Batch Line Parser:

Phases in the batch file line parser:

Phase 0) Read Line:

Phase 1) Percent Expansion:

Phase 2) Process special characters, tokenize, and build a cached command block: This is a complex process that is affected by things such as quotes, special characters, token delimiters, and caret escapes.

Phase 3) Echo the parsed command(s) Only if the command block did not begin with @, and ECHO was ON at the start of the preceding step.

Phase 4) FOR %X variable expansion: Only if a FOR command is active and the commands after DO are being processed.

Phase 5) Delayed Expansion: Only if delayed expansion is enabled

Phase 5.3) Pipe processing: Only if commands are on either side of a pipe

Phase 5.5) Execute Redirection:

Phase 6) CALL processing/Caret doubling: Only if the command token is CALL

Phase 7) Execute: The command is executed

Command Line Parser:

Works like the BatchLine-Parser, except:

Phase 1) Percent Expansion:

- No %*, %1 etc. argument expansion
- If var is undefined, then %var% is left unchanged.
- No special handling of %%. If var=content, then %%var%% expands to %content%.

Phase 3) Echo the parsed command(s)

- This is not performed after phase 2. It is only performed after phase 4 for the FOR DO command block.

Phase 5) Delayed Expansion: only if DelayedExpansion is enabled

- If var is undefined, then !var! is left unchanged.

Phase 7) Execute Command

- Attempts to CALL or GOTO a :label result in an error.
- As already documented in phase 7, an executed label may result in an error under different scenarios.
  - Batch executed labels can only cause an error if they begin with ::
  - Command line executed labels almost always result in an error

Parsing of integer values

There are many different contexts where cmd.exe parses integer values from strings, and the rules are inconsistent:

- SET /A
- IF
- %var:~n,m% (variable substring expansion)
- FOR /F "TOKENS=n"
- FOR /F "SKIP=n"
- FOR /L %%A in (n1 n2 n3)
- EXIT [/B] n